NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

AGENT-BASED TARGET DETECTION IN
3-DIMENSIONAL ENVIRONMENTS

by

Joaquin Steve Correia

March 2005

Thesis Advisor:	Christian J. Darken
Second Reader:	Jeff Crowson

This Thesis Done in Cooperation with the MOVES Institute

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 2005	3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE: Agent-Based Target Detection in 3-Dimensional Environments		5. FUNDING NUMBERS
6. AUTHOR(S) LT Joaquin Steve Correia		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited		12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Visual perception modeling is generally weak for game AI and computer generated forces (CGF), or agents, in computer games and military simulations. Several tricks and shortcuts are used in perceptual modeling. The results are, under certain conditions, unrealistic behaviors that negatively effect user immersion in games and call into question the validity of calculations in fine resolution military simulations. By determining what the computer-generated agent sees using methods similar to that used to generate the human players' screen view in 3-D virtual environments, we hope to present a method that can more accurately model human visual perception, specifically the major problem of a entity "hiding in plain sight".

14. SUBJECT TERMS Perceptual Modeling, Target Detection, Artificial Intelligence, Non-Player Characters, Synthetic Players			15. NUMBER OF PAGES 63
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

AGENT-BASED TARGET DETECTION IN 3-DIMENSIONAL ENVIRONMENTS

J. Steve Correia
Lieutenant, United States Navy
B.A., University of San Diego, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MODELING, VIRTUAL ENVIRONMENTS AND SIMULATION

from the

NAVAL POSTGRADUATE SCHOOL
March 2005

Author:	Lieutenant J. Steve Correia
Approved by:	Christian J. Darken Thesis Advisor
	Jeff Crowson Second Reader
	Dr. Rudolph P. Darken Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Visual perception modeling is generally weak for game artificial intelligence (AI) and computer generated forces (CGF), or agents, in computer games and military simulations. Several tricks and shortcuts are used in perceptual modeling. The results are, under certain conditions, unrealistic behaviors that negatively effect user immersion in games and call into question the validity of calculations in fine resolution military simulations. In games, the most common method of determining whether an agent can observe a target is by casting a line-of-sight ray from the observer to the target (or visa versa). The programmer determines where and how many points represent the observer and the target. For human targets the top of the head is often used. Using this method, a target is detected when the line segment extends from the observer to some predetermined point on the target without intersecting another object in the scene, including terrain. However, if an object, even a small one like a leaf, obstructs this single target point at which the line segment intersects the target, the target is considered concealed. For military simulations more complex models of visual perception exist (for example, the Army's ACQUIRE model). However, inputs to this and similar models still yield questionable results for individual units because of the fixed values often used. For example, in the ACQUIRE model a fixed parameter is used for background contrast, and for the purposes of the calculation, a soldier that is on top of a hill and backlit has the same contrast with a soldier that is in a shaded, wooded area. This is obviously not the case in reality. By determining what the computer-generated agent sees using methods similar to that used to generate the human players' screen view in 3-D virtual environments, we hope to present a method that can more accurately model human visual perception, to enhance the ability of entities to "hide in plain sight".

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First and foremost, I'd like to acknowledge my mother and father, Lucille and Joaquim Correia, for their endless support, love and patience. They individually immigrated to the U.S. and have lived out their own custom-made version of the American Dream. They have always provided me with a supporting environment, but always left me enough room to make my own decisions and mistakes (and occasionally learn from those mistakes) and come to my own conclusions, in my own fiercely independent way. They have always instilled in me the importance of education and a critical mind. My ongoing education is a product of their early and consistent encouragement. For this and more I am forever grateful to them.

I also like to acknowledge Dr. Chris Darken, my thesis advisor, for his eternal patience, great attitude, high level of enthusiasm, and superior knowledge.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM AND THESIS STATEMENT

Visual perception modeling for agents, that is, non-player characters (NPCs) in computer games and computer generated forces (CGF) in military simulations, is generally weak. This has a negative impact on immersion in games and military training simulations and calls into question the validity of analytical military simulations.

We present a novel method that can more accurately model human visual perception by determining what the computer-generated agent sees using methods similar to those used to generate the human players' screen view in 3-D virtual environments. Specifically we deal with the major problem of an entity hiding in plain sight, as described below. A demonstration program build on Delta3D, http://delta3d.org/, an open source simulation engine being developed at the MOVES Institute, is also described.

B. MOTIVATION

Several tricks and shortcuts are used in perceptual modeling. The results are, in many instances, unrealistic behaviors that negatively effect user immersion in games and military training simulations, and call into question the validity of calculations in fine-resolution, analytical military simulations.

In computer games and 3-D graphics-based military simulation, the most common method of determining whether an agent can perceive a target is by casting a line-of-sight ray from the observer to the target (or visa versa). The programmer determines the number of line-of-sight rays that are to be used and where these lines will be traced to on the target. For targets that represent humans, a single line projected to the top of the head is often used. Using this method, a target is detected when the line segment extends from the observer to some predetermined point on the target without intersecting another object in the scene, including terrain. However, if an object, even a small one like a leaf, obstructs this single target point at which the line segment intersects the target, the target is considered concealed. One can easily envision a similar situation for multiple line-of-sight rays.

For military simulations more complex models of visual perception exist, including the Army's ACQUIRE model. However, inputs to this and similar models still yield questionable results for individual units because of the fixed values are often used. For example, in the ACQUIRE model a fixed parameter is used for background contrast and, for the purposes of the calculation, a soldier that is on top of a hill and backlit has the same contrast as a soldier that is in a shaded, wooded area. This is obviously not realistic. Though several efforts have been made to improve this shortcoming, for example see (Champion 99), perception situations are still categorized for the type of landscape and not based on the specifics of the current situation. For example, a fixed contrast is used for a European forest environment, but does not use the actual contrast between an individual soldier and the background. A different environment like a South-American jungle simply uses a different constant.

1. Hiding in Plain Sight

Consider the example of hiding in plain sight. A player might hide within the line-of-sight of the NPC, but use camouflage, smoke, shadow, or clutter to conceal him or herself. Using current line-of-sight models this player would be detected without consideration given to his careful selection of a "hiding" place. This behavior is substantially different than the behavior that we would expect from a human, which would be effected by factors such as camouflage.

C. THESIS SCOPE

This thesis provides a different methodology for providing inputs to a perception system and algorithms for using those inputs. Demonstration software that was developed, based on the Delta 3D open source simulation engine, is also discussed. Issues related to accelerating perceptual calculations using graphics hardware are not covered, though a related thesis along this line is discussed in Chapter II.

D. THESIS ORGANIZATION

The remainder of this thesis is organized into 4 chapters as follows:

1. Chapter II - Related Work

Describes several works related to this thesis, including works in game and military simulation related domains.

2. Chapter III - Algorithms

Describes the algorithms that were developed for this thesis. Also, there is some discussion on how such algorithms might be implemented in more thorough simulations.

3. Chapter IV - Demonstration Software

Describes the demonstration software that was developed for this thesis. The demonstrations software provides a simple proof of concept for the algorithms and ideas presented in the thesis.

4. Chapter V - Analysis and Conclusions

The chapter provides some simple analysis and draws some conclusions for the thesis. Also, suggestions for future work and possible improvements are also provided.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORK

A. INTRODUCTION

There has been a good deal of work done related to perception for intelligent software agents in virtual environments, and most of the more complex and important work has been done in the military simulation world. Pursel provides a great overview of many of the more important military algorithms and technologies. Much of the material on military simulation will be a summary of (Pursel04) that will be provided here for continuity.

B. DETECTION MODELS IN MILITARY SIMULATIONS

1. NVESD ACQUIRE, its Applications and Modifications

Originally, the U.S. Army's Night Vision and Electronic Sensors Directorate (NVESD) ACQUIRE model was developed to predict target acquisition performance for imaging systems operating in the visible, near infrared, and infrared spectral bands. ACQUIRE takes as inputs target characteristics, atmospheric conditions, and sensor operating characteristics to predict probabilities associated with a particular target acquisition task. Specifically, the probability of detecting the target given an infinite amount of time and the time-dependent probability of detection are calculated. ACQUIRE has two modes of operation: target spot detection and target discrimination. Target spot detection calculations are based on signal-to-noise ratio theory and require characterizing the system by either minimum detectable contrast or minimum detectable temperature difference. Target discrimination calculations are based on a two-dimensional Johnson cycle criteria methodology, (Johnson58), and require characterizing the system by either minimum resolvable contrast or minimum resolvable temperature difference, depending on if a visible or infrared result is of interest. The probabilities predicted by the model represent the expected performance of an ensemble of trained military observers with respect to an average target having a specified signature and size (NVESD 1). A more extensive description of the internal workings of ACQUIRE is deferred to the algorithms chapter of this thesis.

Recent military simulations that use the ACQUIRE model, to varying degrees, include the Combined Arms and Support Task Force Evaluation Model (CASTFOREM), the Janus Training Simulation, and the U. S. Marine's Team Tactical Engagement System (TTES). See (Pursel04) for a more detailed description of how these simulations use ACQUIRE.

Several extensions to the basic ACQUIRE model have been made. One of the more significant was by (Champion99). Champion quantifies the effects of vegetation on target acquisition in the context of the ACQUIRE model.

2. Individual Combatants

Reece and Wirthlin (Reece96) describe internal visual and audio models for individual combatant computer generated forces (CGFs), and some typical behaviors that result from these models. For the visual model an ACQUIRE-like model is used that was adapted from (Lind95). Reece and Wirthlin implement these models in the Team Tactical Engagement Simulator (TTES), a 3-dimensional simulation developed by the US Marine Corps using distributed interactive simulation technology. TTES, which later became the Small Unit Tactical Training (SUTT), is a virtual-reality training device that supplements live fire exercises for the Marine Corps. Training is conducted in a realistic combat environment where students, armed with simulated weapons, encounter computer-generated hostile and neutral characters.

C. COMPUTER GAMES, VIRTUAL ENVIRONMENTS AND RELATED TECHNOLOGIES

1. Synthetic Vision

Blumberg describes a method where frames rendered from a synthetic creature's perspective are used as inputs to perception. Blumberg terms this approach "synthetic vision." Blumberg used input frames for creature steering in a virtual, indoor, 3-dimensional world in (Blumberg97) and (Blumberg96). Motion energy (Figure 1) is calculated for each half of the image and the virtual creature (in this case a dog) is able to make course corrections and stay within the hallway boundaries, avoid obstacles, and approach items of interest in a complex "Doom"-like environment.

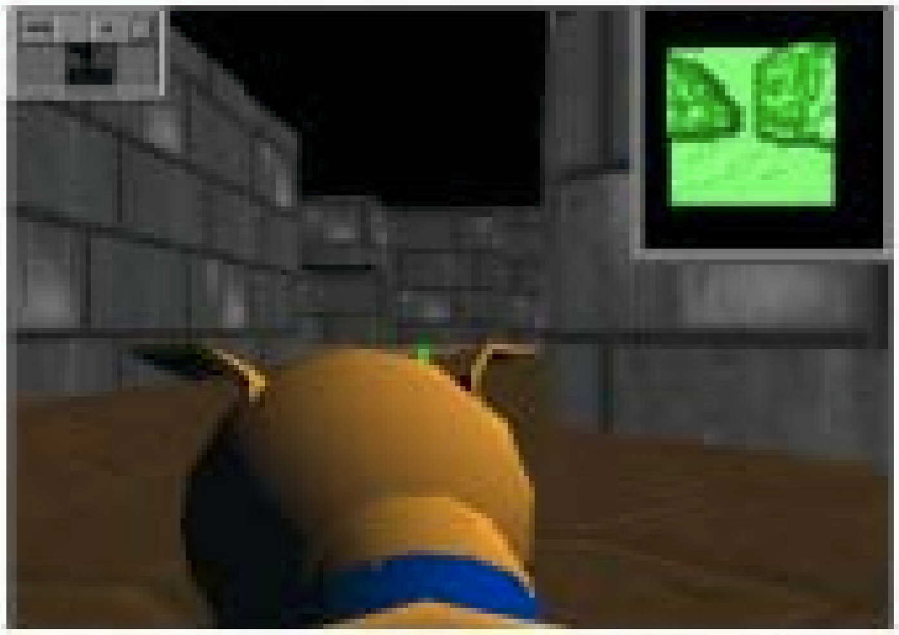


Figure 1. Blumberg's Synthetic Vision

2. Visibility and Concealment Algorithms

Darken describes improved algorithms for determining the visibility of and finding concealment points for agents in 3D environments in (Darken04):

> Many military simulations and computer entertainment products share a need to model the ability of individual entities to see one another and to hide from one another in a 3D virtual environment. The traditional line-of-sight (LOS) visibility model can cause serious problems with hiding behavior. (Page 1)

Algorithms of varying computational complexity include shadow volume, depth map, and sensor grid approaches. A more detailed description of the concealment algorithms as well as an implementation of the sensor grid approach can be found in (Morgan03). The sensor grid approach consists of laying a grid of sensors around the agent that wants to be concealed and then testing the visibility of those sensors from the agent(s) that it wishes to be concealed from. The recommended sensor pattern layout is illustrated in Figure 2.

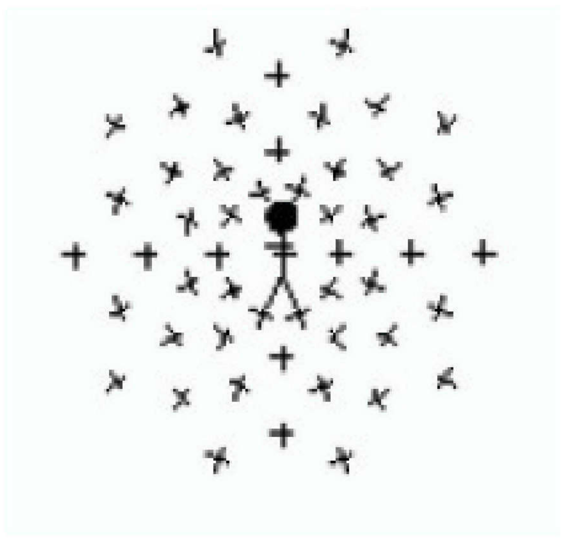


Figure 2. Sensor grid concealment approach

3. Using the Graphics Processing Unit (GPU)

Pursel studied the feasibility of using the GPU for doing perception calculations in (Pursel04). Vertex and fragment shader programs are used to make comparisons of the stored images. All the renders and calculations are performed on the Graphics Processing Unit (GPU) and the result is returned to the agent in reduced form for decision-making. Pursel showed that such calculations can be done on the GPU and, thus, avoid the costs associated with returning large amounts of data from the GPU to main memory. Presently, a good deal of vendor specific code is required to accomplish these calculations on the GPU. Future technologies will likely increase the ease of doing these calculations on the GPU or the cost of returning the full-sized frames to the CPU for calculation.

III. TARGET DETECTION ALGORITHMS

A. INTRODUCTION

In this chapter we describe the algorithms that were developed to more accurately model human visual perception. The Army's Night Vision and Electronic Sensors Directorate (NVESD) ACQUIRE model was chosen as the basis for our visual perceptual modeling. This was done for several reasons. First, ACQUIRE is a model that is often used for modeling visual perception in military simulations. (NVESD1) states that "the ACQUIRE model can be applied to performance prediction of Direct View Optical (DVO) sensors [including the naked eye] but is not recommended for that purpose." In practice it has been used for DVO often (see Chapter II). Also, ACQUIRE is a well-known and widely used model. However, in most implementations many of the important input parameters in ACQUIRE are estimated with fixed values. Our implementation uses specific situational data, taken from a scene render, to a greater extent then has been done previously.

The use of ACQUIRE with the improved input parameters has lead to a perceptual model superior to line-of-sight. However, the basic ACQUIRE algorithm still performs poorly in some situations. These situations will be covered in greater detail below. In this chapter, we present an extension to the basic algorithm that uses by-color contrast and color histograms that improve the algorithm performance in many of these situations.

B. GATHERING DATA FOR COMPARISON, THE USE OF FALSE COLORING

The first step in our algorithm requires the generation of two renders from the perspective of the observing agent: one normal render and one render with any of the targets of interest rendered using flat, false coloring. By false coloring we mean that any targets of interest are rendered in a flat, pre-determined color value that will indicate that it is a subject of the perception. In our demonstration program red is used, that is the RGB (Red, Green, and Blue) values are set to 1.0, 0.0, and 0.0 respectively. The effect is demonstrated in Figure 3.

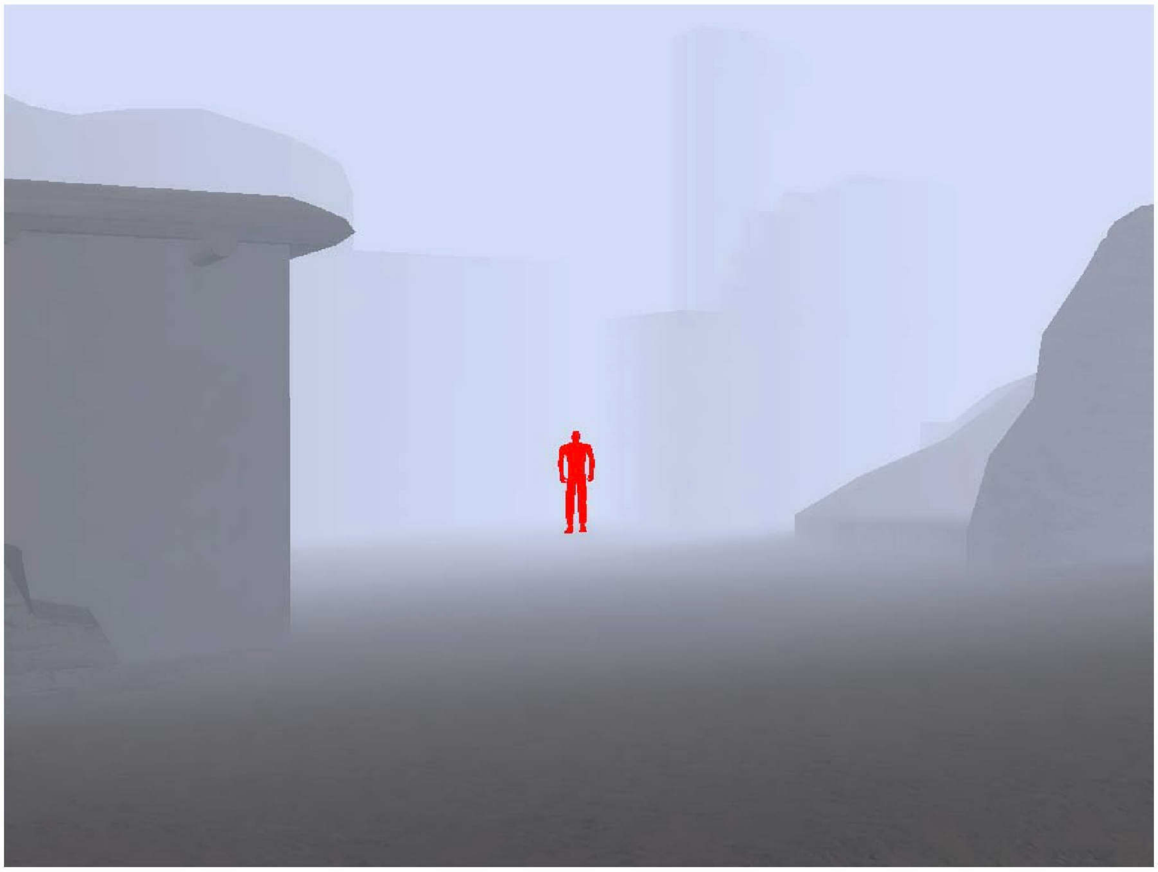


Figure 3. Figure rendered with false coloring

In gaming and training applications, measures will be taken to keep this render from going to the screen. There are several methods for doing this, including rendering to texture. However, for the sake of code and software execution clarity, this off-line rendering was not implemented in the demonstration program. Once the two images are generated they are further processed as follows:

1. The Line of Sight Check

At this point we can easily check for line of sight. If the false color render produced any pixels with false coloring we know that line-of-sight exists. If no false-color pixels were rendered, line-of-sight does not exist and we can exit with no probability of detection.

2. The Mini-Render

Next we reduce the size of the images to include just the target and the area surrounding the target. We call this the mini-render, Figure 4.

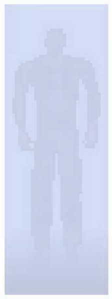


Figure 4. A Mini-Render

There are several reasons for doing this. First and foremost, the area directly surrounding the target has the greatest bearing on how well hidden the target is. The algorithms that are used compare the target and non-target pixels values. Often pixels that represent the target are only a small number of the pixels in a given render. If all the non-target pixels in the scene are used one might obtain average values that are not indicative of the area surrounding the target.

For example, imagine a figure in camouflage hidden in a heavily shadowed forest with a washed out hazy sky above. In this situation the hazy sky would greatly effect the pixel values for the background, but, in reality, would have little bearing on whether the target stands out in the scene or not. Another, example is a figure silhouetted in a doorway at night. If the scene were taken in its entirety it is possible that we would calculate very little contrast between the figure and the background. However, given that the figure is backlit in a doorway any observer would easy detect him/her.

C. THE ACQUIRE-LIKE ALGORITHM

ACQUIRE was used as the basic algorithm for calculating detection for the reasons stated above. Our algorithm, as implemented, provides the following: the probability of detection of the target given an infinite amount of time (i.e. the time-independent detection probability) and the time-dependent probability of detection (i.e. the probability that a target will be detected before a given number of seconds). Our algorithm combines the two operating modes of ACQUIRE: target spot detection and target discrimination.

The first input to ACQUIRE is the target signature. For visual systems this consists of the contrast difference between the background and the target. ACQUIRE starts with the inherent contrast, that is, the contrast with no degradation for atmospheric or combat-induced conditions, like smoke, fog, reflectance, etc. However, in the virtual environment these factors are represented graphically; thus we calculate the contrast directly and do not need to estimate the effect of these obscurants using the ACQUIRE empirical equations.

To calculate the contrast for our basic ACQUIRE implementation we take the weighted average of the pixels in the mini-render that are part of the target, p_t, and that are part of the target background, p_b. However, ACQUIRE defines the contrast by the irradiance of the target and background in units of watts per meters square (W/m^2). The units that we have in the virtual environment are pixel values. As it turns out there is a relationship between the units that AQUIRE uses and the pixel values we have, though the relationship is not a linear one.

Watts per meter squared is proportional to lumens as follows:

$$lumen = \frac{1}{683} Watts @ 555nm \qquad (1)$$

Also, lumens are related to Lux as follows:

$$lux = \frac{lumen}{m^2} \qquad (2)$$

If one ignores the color dependence inherent in lumens we can conclude that Lux is proportional to Watts per meter squared:

$$Lux \propto \frac{watts}{m^2} \qquad (3)$$

There is a known relationship between Lux and pixel value. It is a squared relationship and thus,

$$\frac{watts}{m^2} \propto pixel\_value^2 \qquad (4)$$

pixle_value is numeric value a given pixel. Given the results of these calculations, we now are able to use pixel values to calculate the contrast. Thus, we are able to substitute the pixel value squared in for the irradiances of the target and background. Calculation of contrast becomes:

$$C = \left| \frac{p_t^{\,2} - p_b^{\,2}}{p_b^{\,2}} \right| \qquad (5)$$

C is the contrast, p_t is the average pixel value of the target and p_b is the average pixel value of the background. To calculate the p values we average the R, G, and B values for each pixel over either the target or background, as appropriate, within the mini-render.

Next we need some measure of the angle, θ, which the angle that the target occupies in the field of vision of the observer, Figure 5.

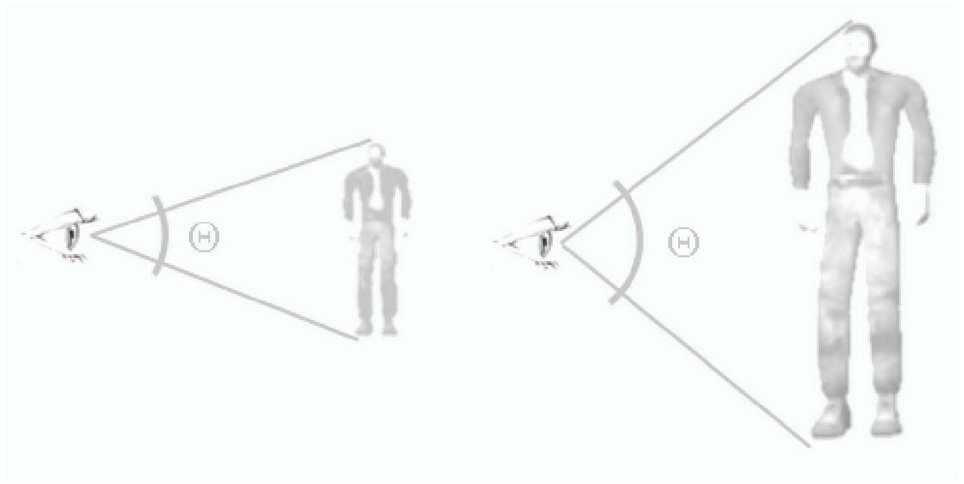


Figure 5. Relationship between θ and the number of pixels used to represent the target

In ACQUIRE this is estimated with the ratio of the critical dimension (CD), i.e. the square root of the exposed surface area of the target, and the range. This yields an approximation of θ. We estimate this parameter to within a constant factor using the square root of the number of target pixels that can be seen by the observer, num_of_pixels, a piece of information that is readily available in our graphical environment.

$$\theta \propto \sqrt{num\_of\_pixels} \qquad (6)$$

Next ACQUIRE calculates the number of resolvable cycles, N. We calculate N as follows:

$$N = \left| \frac{p_t^{\,2} - p_b^{\,2}}{p_b^{\,2}} \right| \cdot \sqrt{num\_of\_pixels} \qquad (7)$$

Now we use the above calculations as inputs to the rest of the ACQUIRE equations. We calculate the time-independent probability of detection as follows:

$$P_{\det} = \frac{(N/N50)^E}{1+(N/N50)^E} \qquad (8)$$

$$\text{where } E = 2.7 + 0.7(N/N50) \qquad (9)$$

We use a default value of 1.0 for N50, as suggested in (Lind95) and (Reece96). N50 is the number of cycles required to be resolved on the target in order to achieve a 50% probability of discrimination (NVESD1). In other words, N50 is the number necessary for half of a group of trained observers to detect a given target, on average, under certain lighting conditions. We believe that a better estimation for N50 can be derived from experimentation, but such is beyond the scope of this study.

We further calculate the time dependent target detection as

$$P_t = P_{\det} \cdot (1 - e^{-t/(m \cdot \tau)}) \qquad (10)$$

Where $P_{\det}$ is the time-independent probability of detection calculated above, t is the length of time of interest, m is the number of sensor fields in the field of regard (for our software we assume a value of 1) and τ is the mean detection time in seconds for all observers that eventually detect the target. The field of regard (FOR) is the size of the area that an observer is responsible for scanning. The FOR can be a few degrees up to 360 degrees. Experimental data indicates that the value of τ is related to the overall difficulty of detecting the target (NVESD1). ACQUIRE uses an approximation for **τ**, which we adopt, that is given by

$$\tau = \frac{6.8}{N/N50}\sec \qquad (11)$$

The result produced by our algorithm solves several of the problems with the simple line-of-sight calculations used in many applications as a substitute for perception. For example, it easy takes into account fog. Fog has the effect of dimming the contrast between a target and the background. In dense fog there may be no contrast at all. Figure 6 depicts this situation. Figure 7 shows the same scene, but with the target in false coloring.

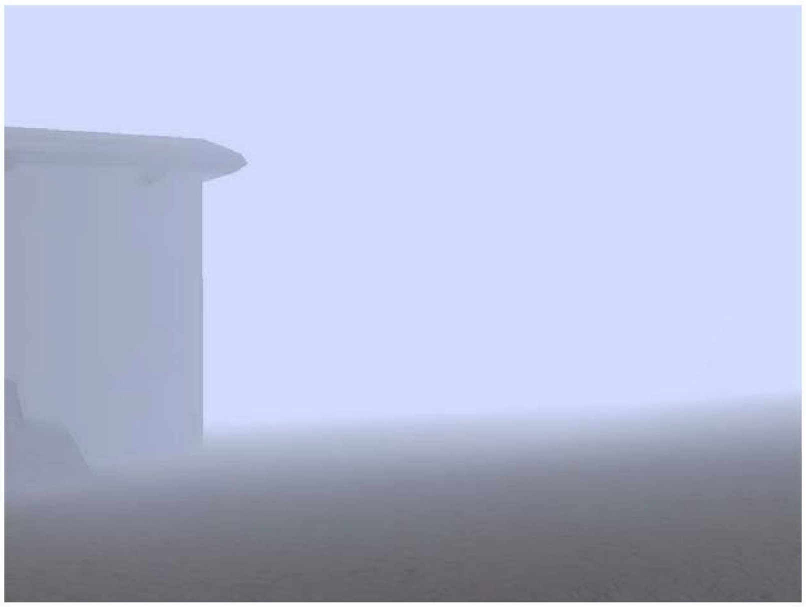


Figure 6. Foggy Scene with line-of-sight to the target, but the target is undetectable

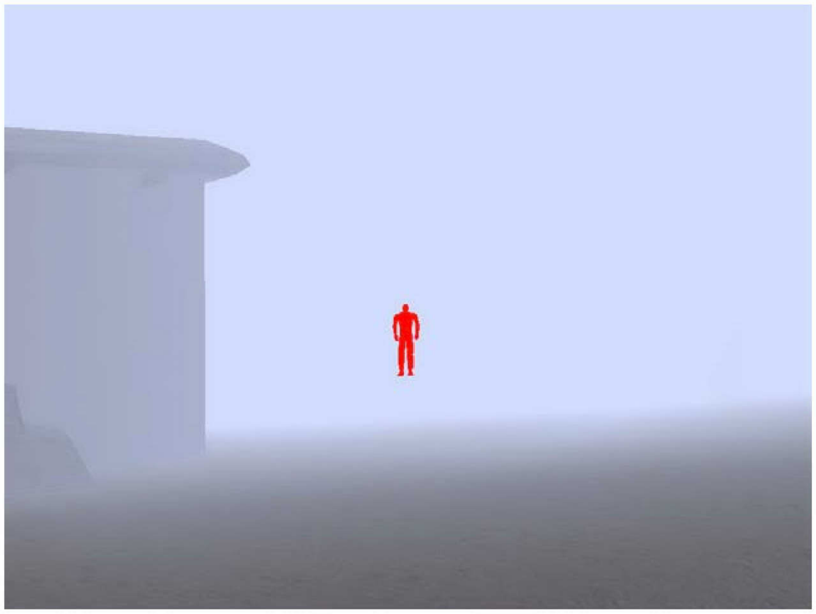


Figure 7. False color version of Figure 6

If line-of-sight were used in this situation the target would have been detected because line-of-sight does exist. Clearly, however, the target should not be visually detected.

One weakness of the basic ACQUIRE algorithm is that we use pixel values for the target and background are averaged over all the colors. Because of this there is the possibility of an error given the correct situation. Imagine a figure wearing a red suit in a green forest. Obviously, any non-color blind observer would spot this anomaly rather quickly. However, if the red color of the suit and average green color of the forest are equal (assume the other colors are the same), they will average out to the same value and the algorithm would notice no contrast between the target and the background and, thus, return a zero probability of detection. Obviously, this is incorrect. Modifications to the above algorithm that alleviate this problem are presented below.

D. MODIFICATIONS TO THE ACQUIRE-LIKE ALGORITHM

1. The By-Color ACQUIRE-Like Algorithm

Rather than taking the average pixel value for the target and the background, we compute contrast for each color (red, green, and blue) and calculate the overall contrast as follows:

$$C = \frac{1}{3}\left|\frac{{p_{t\_red}}^2 - {p_{b\_red}}^2}{{p_{b\_red}}^2}\right| + \frac{1}{3}\left|\frac{{p_{t\_green}}^2 - {p_{b\_green}}^2}{{p_{b\_green}}^2}\right| + \frac{1}{3}\left|\frac{{p_{t\_blue}}^2 - {p_{b\_blue}}^2}{{p_{b\_blue}}^2}\right| \tag{12}$$

We give an equal weighting to each of the three component colors. Equation (12) solves the problem of the red target in a green forest presented above.

2. Color Histograms

We have also implemented a version of the calculations that uses color histograms to compute the contrast. There is one histogram for the target and one for the background and for each color: red, blue and green. This yields a total of 6 histograms. Our implemented version uses 8 color bins, which was an arbitrary choice. We then innumerate through the mini-render and count the number of pixels within each of the pixel intensity ranges, according to Table 1.

Bin Index	Minimum Pixel Intensity	Maximum Pixel Intensity
0	0	31
1	32	63
2	64	95
3	96	127
4	128	159
5	160	191
6	192	223
7	224	255

Table 1. Pixel value ranges that cause counts within certain bins

Figure 8 illustrates a histogram for the red pixel values of a target and Figure 9 illustrates the histogram of the red pixel values of the background. Note that the number of pixels will vary between the target and the background.

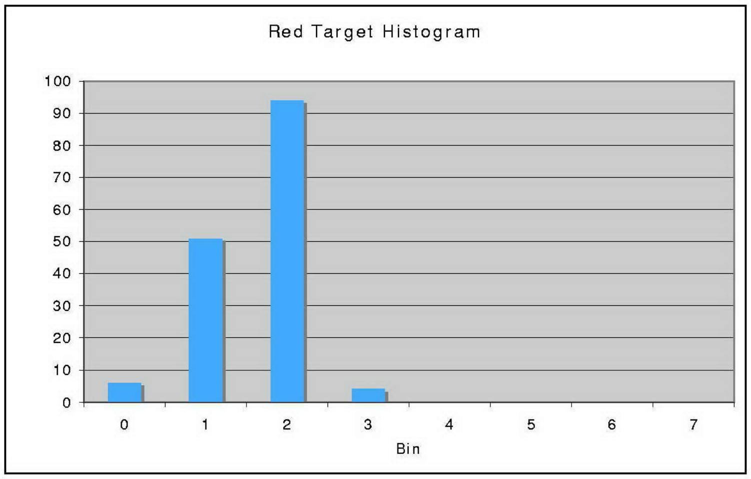


Figure 8. Histogram for the Red Pixel Values of a Target

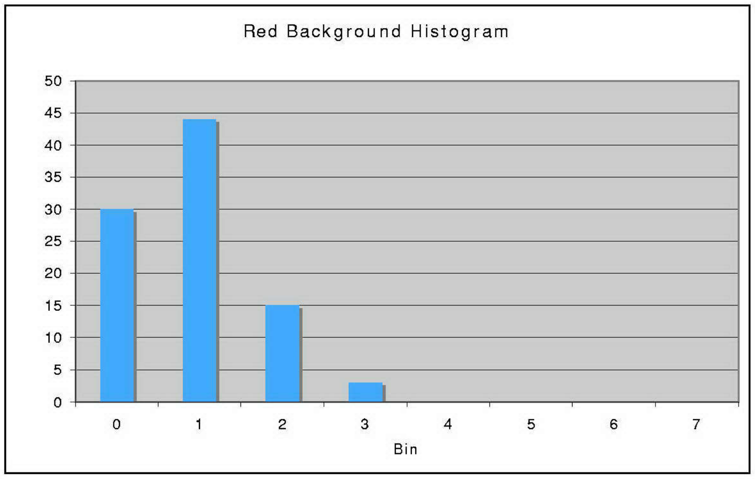


Figure 9. Histogram for the Red Pixel Value of the Background

Once the pixel iteration is complete the histograms are converted to probability mass functions (PMFs), that is, the sum of all the bins within the histogram are equal to one. We then calculate the L1 difference, i.e. the sum of the differences between the background and target for each bin, Figure 10.

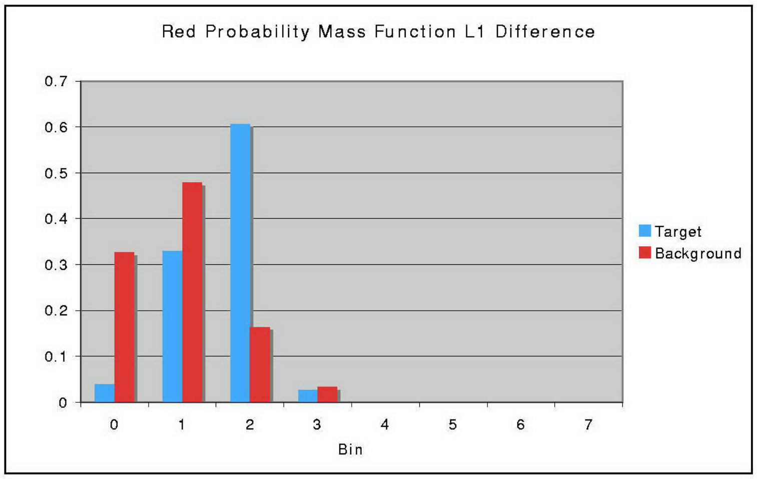


Figure 10. The sum of the differences between the target and background counts within each bin are used to calculate the L1 difference

Equation (13) is used to calculate the overall contrast using the PMFs given all 6 histograms.

$$C = \frac{1}{3}\sum_{bins}\left|trgtR_i - bckR_i\right| + \frac{1}{3}\sum_{bins}\left|trgtG_i - bckG_i\right|\frac{1}{3}\sum_{bins}\left|trgtB_i - bckB_i\right| \qquad (13)$$

The result of the contrast calculation is used as the contrast input to the remaining calculations above.

IV. DEMONSTRATION SOFTWARE

A. INTRODUCTION

This chapter gives an overview of the demonstration software developed in conjunction with this thesis. Our software is built with the Delta 3D open-source simulation engine, which is actively under development at the MOVES Institute at the Naval Postgraduate School in Monterey, CA. The goal of the demonstration software is twofold: First, the software provides a proof of concept for the basic idea put forth in this thesis, i.e., renders from a given perspective can be used as inputs for perception algorithms. Additionally, our software was used in this thesis and can be used in the future as a test bed for the development and testing of perception algorithms.

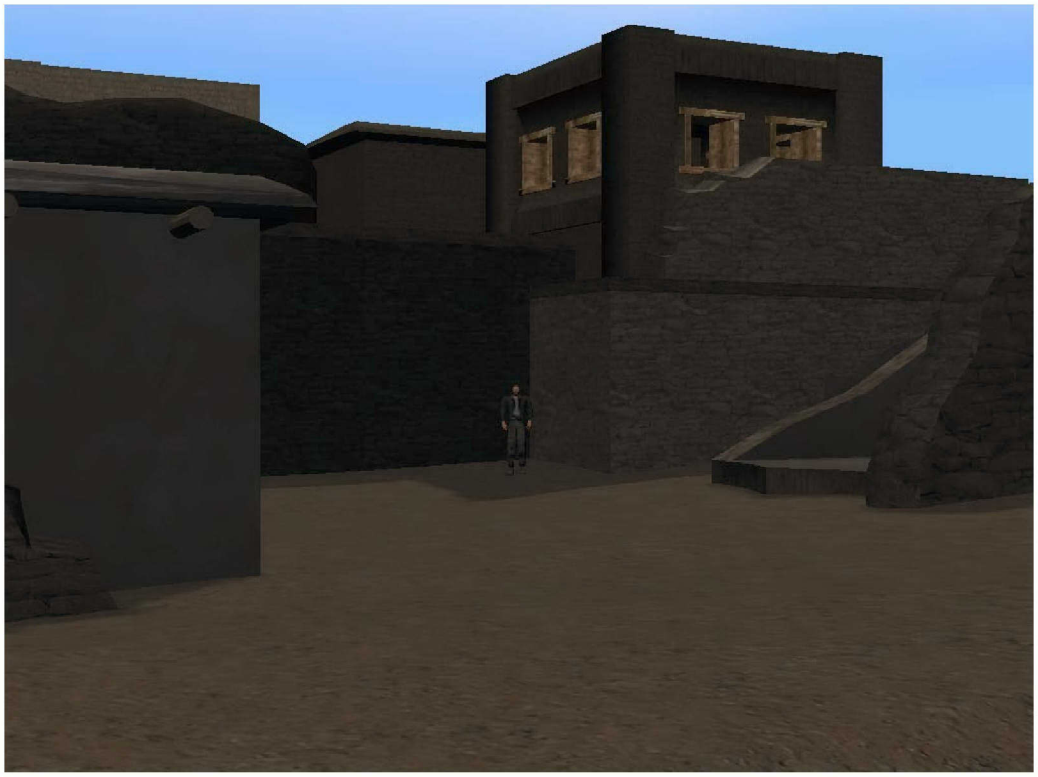


Figure 11. A scene from the demonstration software

Figure 11 portrays a scene in a town with one human-like figure. In our software the human figure acts as the target for the algorithms.

B. FEATURES

All the features in the software are controlled with keyboard input. The user can control several aspects of the scene using the keyboard including the position and orientation of the figure, the amount of visibility into the fog (when on) and the time of day. Additionally, the user can initiate the execution of the algorithms at any time.

1. Figure Position and Orientation Control

Figure position and orientation are changed using the left and right arrow keys and the "W" key. "W" causes the figure to walk forward. The combination of Shift and "W" cause the figure to run forward. The right arrow key causes the figure to turn right and the left arrow key causes the figure to turn left.

2. Fog Control

Visibility through the fog within the scene is controlled using the plus (+) and minus (-) keys. Pressing the plus key increases the visibility in the scene. If visibility reaches beyond 100 the fog is turned completely off (until the minus key is pressed again). The minimum visibility is 1. Visibility changes in increments of 10.

3. Time of Day Control

Time of day is controlled using the number pad. Pressing a given number will change the time of day within the scene as follows:

0	0200
1	0400
2	0600
3	0800
4	1000
5	1200
6	1400
7	1600
8	1800
9	2000

Table 2. The number key and corresponding time of day combinations in our demonstration software

4. Executing Perception Calculations

Perception calculations are executed by pressing the "S" key. Output is sent to the command-line console. The user must ensure the figure is stationary during percept calculations. Code was not included to freeze the scene during screen captures. If the figure is moving during a capture the false color and normal color frames will not coincide and any calculations will not be accurate.

C. IMPLEMENTATION

The goal of this section is to give a brief overview of the class structure of the demonstration program. The demonstration program is implemented using the Delta 3D open source gaming and simulation engine (Delta3D). Delta 3D is written in the C++ programming language and developed in Microsoft Development Environment, Version 7.1. Delta 3D includes many easy to use classes to make simulation development easier. These classes were used or extended whenever possible. Additional open source code was used and credit is given below.

1. Selected Classes and Methods

The program uses the following classes and methods. A brief discussion of each is provided:

<u>Demo class</u>. This is a class derived from the Delta 3D Application class. It is the base level class for the application and it contains the basic components required for the application including the application window, scene camera, and simulation loop. This class is responsible for loading objects into the scene as well as application flow control. It also allows for alterations to or calculations on the scene to occur at a specific point in the render process, i.e., in the post-render phase (but before the next render). Lastly, the control keys for the program are mapped in this class.

<u>SyntheticPerception class</u>. This class is responsible for doing all the perception calculations. The methods that produce the probability of detection calculations take 2 images as arguments: one normal image and one image with a false colored target. The calculations described in chapter 3 are implemented in this class.

CharController class. Controls target movement in the software. Also controls the false coloring of the target.

Image class. This is a utility class that takes care of many of the details associated with image access. Once screens are read from the frame buffer, an Image object is generated from the character array. This class was written by Chris Darken.

Bitmap utilities. This is a set of bitmap utilities written by Michael Sweet. It allows OpenGL-compatible images to be written to and from BMP (bitmap) files.

V. ANALYSIS, CONCLUSIONS, AND SUGGESTED FUTURE WORK

A. INTRODUCTION

Each of the perception algorithms discussed in this thesis has some advantages and disadvantages, which we will discuss in more detail in this section. Also, we provide some cursory comparisons between the outputs of the algorithms versus some experimental data that was gathered during a pilot study. Finally, we will provide some conclusions, insights, and suggestions for future work.

B. ANALYSIS OF ALGORITHMS

1. Line-of-Sight Algorithms and Graphics Pipeline Issues

Many of the disadvantages and shortcomings of line-of-sight algorithms have been previously described in this thesis. These included the inability to deal with fog and camouflage, i.e., situations were there is very little contrast between the target and the background. Additionally, problems may occur if only a small part of the target is obstructed by an obstacle, yet it may include that point to which the line-of-sight rays are traced. In other words, it runs the risk of not detecting a mostly visible target.

However, there are some advantages to using line-of-sight calculations. First and foremost it can be fast, especially when the number of line-of-sight calculations is small. Also, it holds an advantage over the paradigm presented in this thesis in that no data must be read back from the graphics pipeline. With current hardware this is a very costly procedure. In effect, all traffic on the graphics pipeline must be stopped to allow for the return data, which can be substantial.

There are two possible solutions to this limitation. The first solution was presented in (Pursel04). Pursel demonstrated that perception calculations can be executed on the Graphics Processing Unit (GPU) and with only a result being returned to main memory, thus alleviating the need to return large chunks of data. General Purpose Computing on the GPU is an active area of interest and research because of the ever-increasing speed and highly parallel nature of the GPU. The GP2 conference (GP2_04) is a forum for such research and discussion. However, this approach also has its problems: First, many of the mechanisms for carrying out such calculations, i.e., Application Programmer Interfaces (APIs), are vendor specific. Also, most of these APIs are not well documented. Some standards have been developed recently, but there is still no clear single standard API for GPU programming. Much of the programming community seems to be waiting for a single standard before venturing onto the GPU.

The other possible solution to the graphics pipeline issue is the ever increasing bandwidth from the graphics pipeline back to main memory. PCI Express, the new input/output bus standard, promises precisely this increase. In the near future, reading back pixel data will likely be a less bus-intensive procedure.

2. Determining Line-of-Sight Using False Color

Determining target line-of-sight using false color is a trivial matter: If there are any false-color pixels in the scene, line-of-sight exists. In fact, an even greater amount of information is easily available. For example, if the scene is rendered with nothing but the target (no buildings, etc) one can easily determine precisely the percentage of the target that is exposed, not just the estimate provided by multiple line-of-sight rays. Of course, caution must be taken when using false color to ensure that the designated false color is not used anywhere else in the scene. If this is the case, any calculations would be invalidated.

3. The ACQUIRE-Like Algorithm

The ACQUIRE-like algorithm introduces several advantages over line-of-sight rays and line-of-sight using false coloring. The most obvious example is its ability to deal with fog. In fog, a clear line-of-sight may exist, but the target may still be completely undetectable. The ACQUIRE algorithm deals with this by comparing the target contrast with the background contrast. Essentially, if the average contrast of the target and background are similar then a target is difficult to detect. Intuitively this makes perfect sense. However, this is not a solution for all cases (see section B below).

ACQUIRE does not deal with issues like clutter, texture and continuity. These are all factors that effect detection likelihood in a human. Also, the averaging of colors that ACQUIRE requires can have some undesirable consequences (as pointed out at the end of Chapter 3).

4. Improving the ACQUIRE-Like Algorithm Using Color Components and Color Histograms

We dealt with the problem of averaging colors in two ways: averaging each color individually (then taking the weighted average) and using color histograms. These methods solve the averaging problem, but still leave unresolved the other issues of clutter, texture, etc.

C. EXPERIMENTAL DATA AND RESULTS

In this section we provide some comparison and comments between the algorithms presented in this thesis and a pilot perception experiment conducted at the Naval Postgraduate School.

1. The Pilot Experiment

We conducted a target detection pilot experiment at the Naval Postgraduate School (NPS) in which fourteen subjects were seated in front a large screen display and static images were displayed to them as shown in Figure 9. All subjects were students at NPS and had varying military backgrounds and experiences.

Figure 12. A subject receiving instructions for the pilot experiment

A total of 37 images were displayed, the first 4 being example images that were used to indoctrinate subjects into the detection task. For the 33 remaining screens subjects were asked to search the screen for a single human-like target. Subjects would indicate that they had detected the target by left clicking on the target using a mouse. Subjects were asked to find the target as quickly as possible, but to take all the time they needed. A small number of the screens had no target and the subjects could indicate that they perceived this by performing a different action. Each user was given the same scripted instructions at the beginning of the experiment. Also, users filled out questionnaires after the experiment that disclosed what level of experience they had with first-person shooter games, in general, and the games used for the screen shots, specifically. They had a wide range of experiences. There did not appear to be any correlation between game experience and detection times.

2. Selected Screen Shots: Experimental vs. Algorithmic Results

Here we present a sampling of the images used in the pilot experiment, the results from those experiments, and the results from processing by our algorithms. This will both highlight the strengths and weaknesses of the approach we have taken and argue for further research in the area of perceptual modeling. It should be noted that ACQUIRE is relatively well adapted to the types of scenes presented in our data set. The issues regarding averaging color are not represented here. Only the images with the more interesting results are included.

3. Screen Shot 06

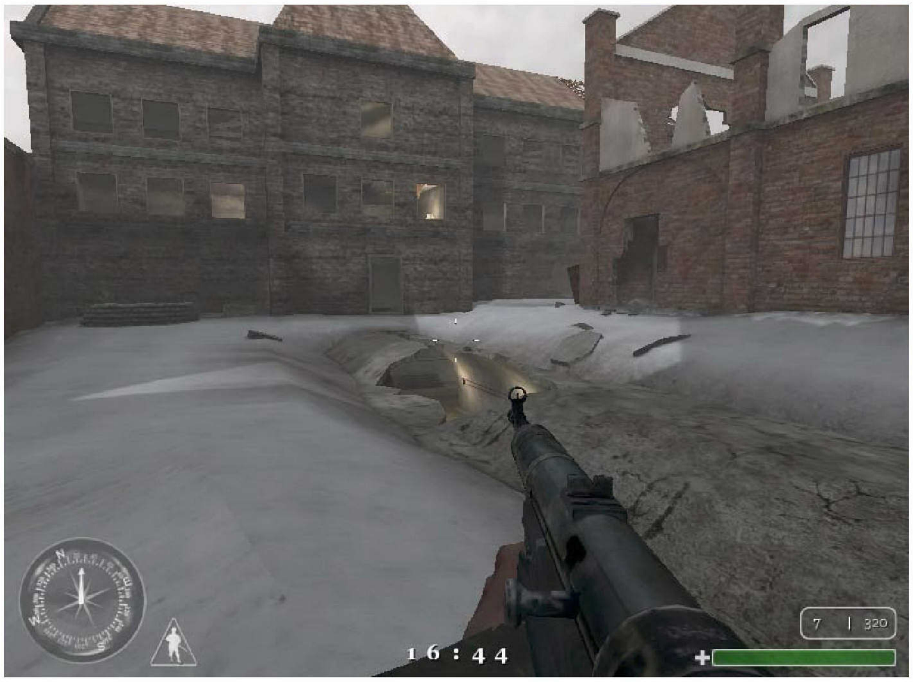


Figure 13. Screen Shot 06

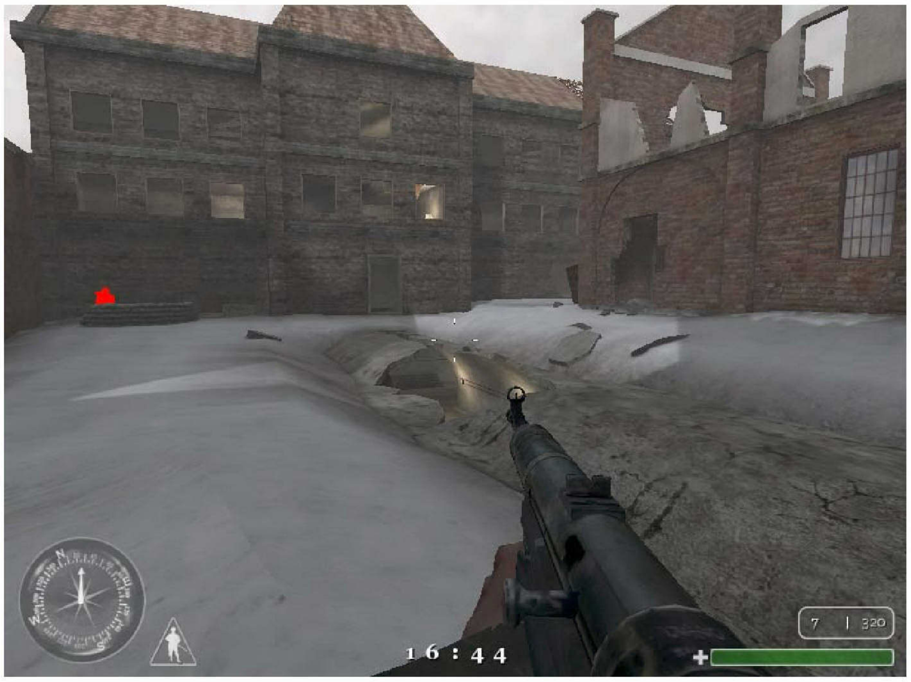


Figure 14. Screen Shot 06 with false coloring

Figure 13 depicts Screen Shot 06, as displayed to subjects in the pilot experiment. Figure 14 shows the same screen with the target in false coloring (to aid in target detection by the reader). The false coloring of the targets in these images was done by hand using Adobe Photoshop. The results of the pilot experiment for this image were as follows:

- Of the 14 subjects 8 were able to detect the target in this scene, a 0.5714 probability of detection
- Of the 8 that successfully detected the target 4 did so within 5 seconds, thus a 5-second, time-dependent probability of detection of 0.2857. Detection times were distributed as per Figure 15 below:

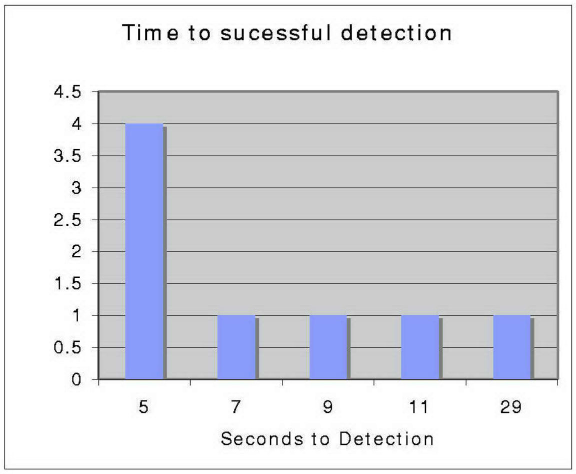


Figure 15. Distribution of successful experimental detection times for Screen Shot 06

Results from executing the perception algorithms on Screen Shot06 yielded the following results:

- The probability of detection for the ACQUIRE-like algorithm was 0.6545, with a 0.3832 probability of detecting the target within 5 seconds.
- The probability of detection for the Color modified ACQUIRE-like algorithm was 0.6577, with a 0.3860 probability of detecting the target within 5 seconds.
- The probability of detection for the Color histogram algorithm was 0.5746, with a 0.3170 probability of detecting the target within 5 seconds.

Table 3 below is provided for easier data comparison:

Source	Time-independent Probability of Detection	Probability of Detection within 5 Seconds
Experiment	0.5714	0.2857
ACQUIRE-like algorithm	0.6545	0.3832
By-Color ACQUIRE	0.6577	0.3860
Color Histograms	0.5746	0.3170

```
Processing Frames . . .

Using the ACQUIRE-like algorithm
        Ave. Target Pix Val: 42.8177  Ave. Bckgrd Pix Val: 44.8614
        Contrast Value:      0.0890382
        n:                   1.19789
        e:                   3.53852
pdet: 0.654507
p(5.0): 0.383246

Using the by-color ACQUIRE-like algorithm
        Ave. Target Red Val: 44.4309  Ave. Bckgrd Red Val: 45.8427
        Ave. Target Grn Val: 43.1492  Ave. Bckgrd Grn Val: 45.1685
        Ave. Target Blu Val: 40.8729  Ave. Bckgrd Blu Val: 43.573
        Contrast Value:      0.0893846
        n:                   1.20255
        e:                   3.54178
pdet: 0.657743
p(5.0): 0.386073

Using color histograms
Target Red:   [0      178     3       0       0       0       0       0]
Bkgrnd Red:   [0      80      9       0       0       0       0       0]
Target Green: [2      177     2       0       0       0       0       0]
Bkgrnd Green: [0      80      9       0       0       0       0       0]
Target Blue:  [7      173     1       0       0       0       0       0]
Bkgrnd Blue:  [0      80      9       0       0       0       0       0]
        L1 diff:             0.0810665
        n:                   1.09064
        e:                   3.46345
pdet: 0.574564
p(5.0): 0.316896
```

Table 3. Detection probability comparison table and program output for Screen Shot 06

We believe that the algorithm yields reasonable results for Screen Shot 06, as this Screen Shot was difficult for the experimental subjects because of the low level of contrast between the target and the background. Contrast is one of the main inputs for the ACQUIRE model, and, our ACQUIRE-like algorithm was able to handle it in a reasonable manner. For example, consider a line-of-sight calculation on this target. The target would immediately be detected in most cases. However, almost half of the experimental subjects were unable to detect the target. Thus, in this case our ACQUIRE-like algorithm displays a desirable, human-like behavior that a normal line-of-sight calculation would not.

4. Screen Shot 17

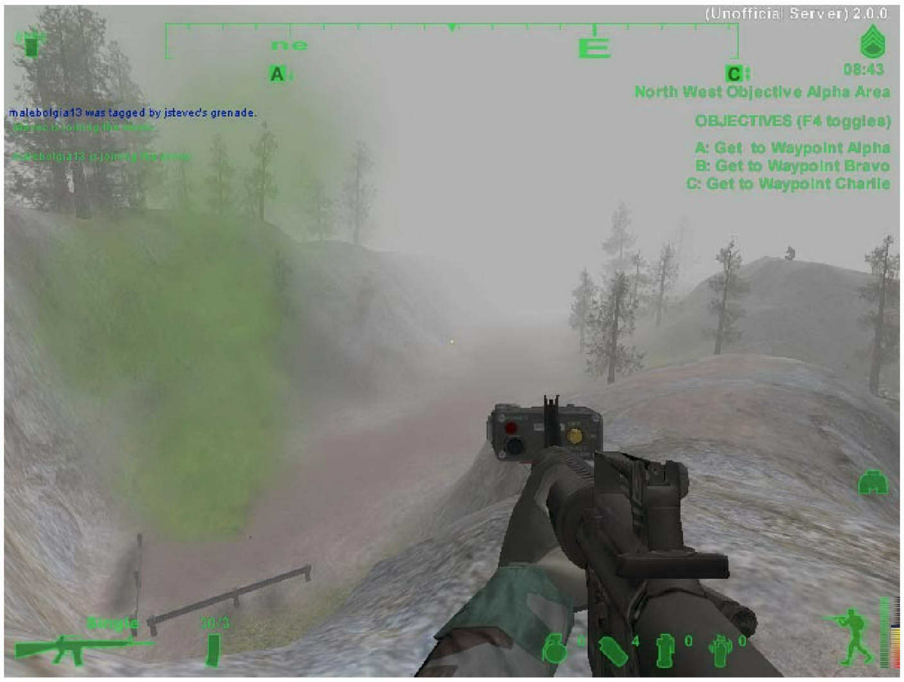

(Unofficial Server) 2.0.0
08:43
North West Objective Alpha Area
OBJECTIVES (F4 toggles)
A: Get to Waypoint Alpha
B: Get to Waypoint Bravo
C: Get to Waypoint Charlie
malebolgia13 was tagged by jsteved's grenade.

Figure 16. Screen Shot 17

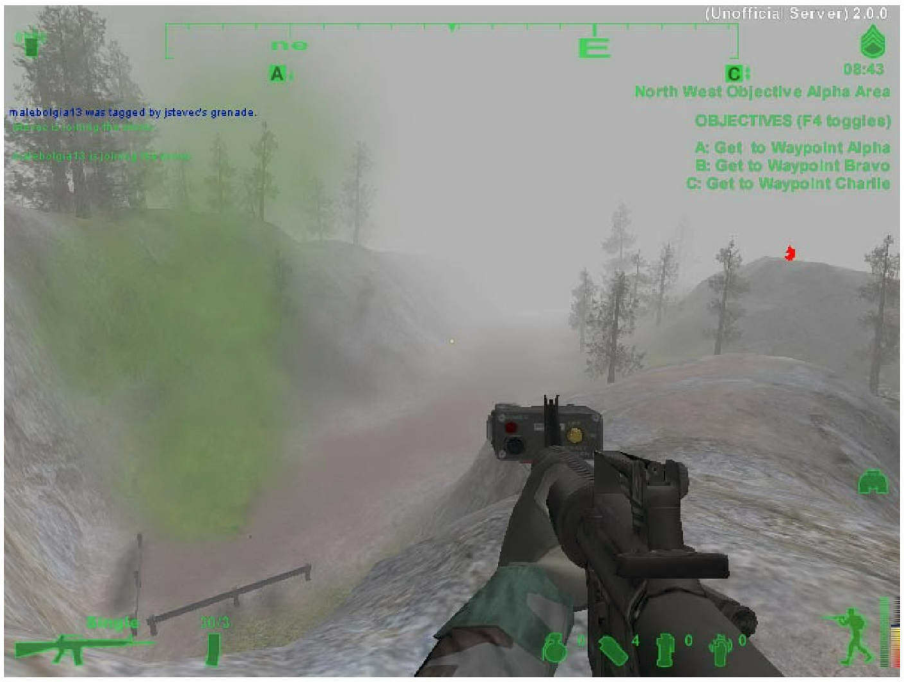

(Unofficial Server) 2.0.0
08:43
North West Objective Alpha Area
OBJECTIVES (F4 toggles)
A: Get to Waypoint Alpha
B: Get to Waypoint Bravo
C: Get to Waypoint Charlie
malebolgia13 was tagged by jsteved's grenade.

Figure 17. Screen Shot 17 with false coloring

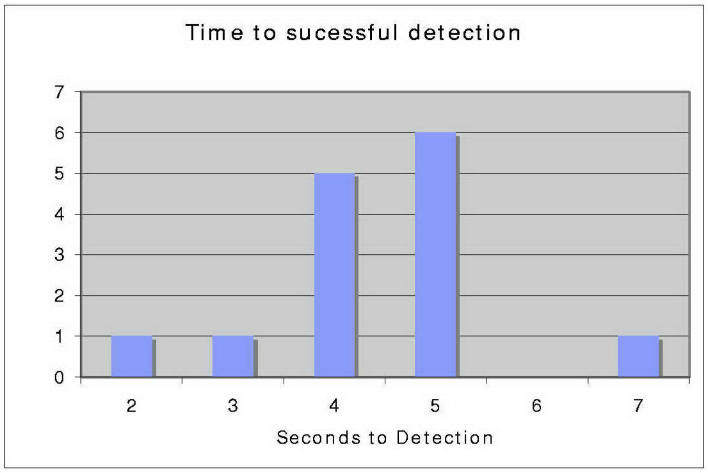


Figure 18. Distribution of successful experimental detection times for Screen Shot 17

Source	Time-independent Probability of Detection	Probability of Detection within 5 Seconds
Experiment	1.0	0.9286
ACQUIRE-like algorithm	0.9998	0.9470
Color-modified ACQUIRE	0.9998	0.9470
Color Histograms	1.0	0.9472

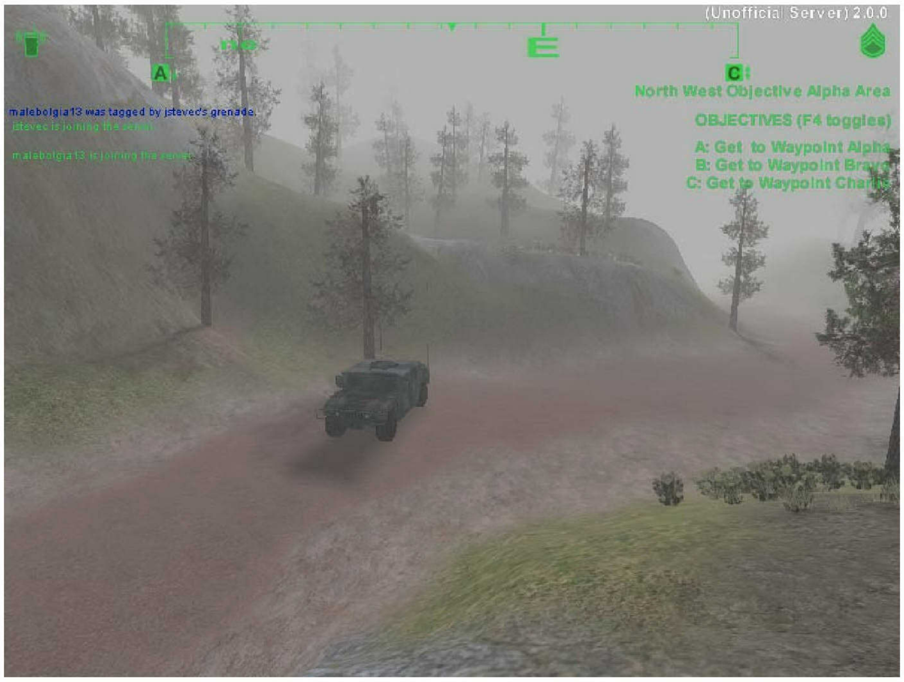
```
Processing Frames . . .

Using the ACQUIRE-like algorithm
        Ave. Target Pix Val: 112.84    Ave. Bckgrd Pix Val: 164.732
        Contrast Value:      0.530783
        n:                   4.47246
        e:                   5.83072
pdet: 0.999839
p(5.0): 0.947044

Using the color-aware ACQUIRE-like algorithm
        Ave. Target Red Val: 113.197   Ave. Bckgrd Red Val: 164.761
        Ave. Target Grn Val: 113.254   Ave. Bckgrd Grn Val: 164.761
        Ave. Target Blu Val: 112.07    Ave. Bckgrd Blu Val: 164.674
        Contrast Value:      0.530774
        n:                   4.47238
        e:                   5.83067
pdet: 0.999839
p(5.0): 0.947044

Using color histograms
Target Red:   [0      0       0       70      0       1       0       0]
Bkgrnd Red:   [0      0       0       0       5       41      0       0]
Target Green: [0      0       0       70      0       1       0       0]
Bkgrnd Green: [0      0       0       0       5       41      0       0]
Target Blue:  [0      0       0       70      0       1       0       0]
Bkgrnd Blue:  [0      0       0       0       5       41      0       0]
        L1 diff:             0.887324
        n:                   7.47672
        e:                   7.93371
pdet: 1
p(5.0): 0.947196
```

Table 4. Detection probability comparison table and program output for Screen Shot 17

For Screen Shot 17, depicted in Figure 16, we once again obtain a reasonable result. However, this shot has different properties from the previous screen. Here we find that the target is nicely silhouetted against the background, though some attenuation due to haze is depicted. Additionally, the target is at a substantially greater distance from the observer. In this case the contrast difference was substantial and negated the small number of pixels representing the target.

From a human perception perspective there is another possible reason for the high detection rate. The contrast is important, but so is the fact that the target is situated on top of a hill that is otherwise smooth. That being said, this is not a fact that is easily quantifiable for input to a perception algorithm. Some techniques do exist, such as those in the field of computer vision, as in (Forsyth03). Using some of these techniques has the potential to further improve perception algorithms.

5. Screen Shot 18


(Unofficial Server) 2.0.0

North West Objective Alpha Area

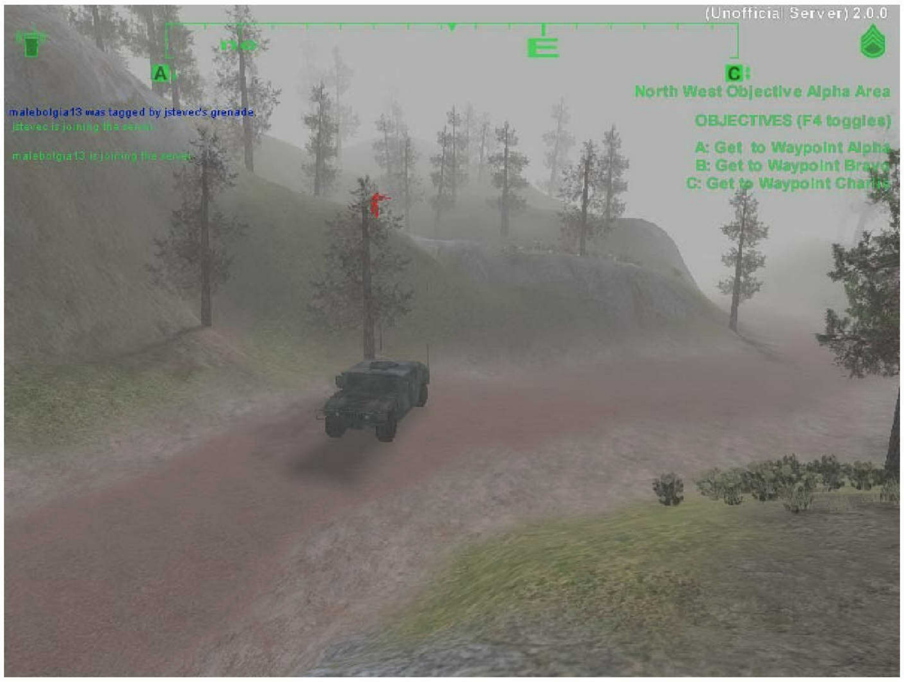
OBJECTIVES (F4 toggles)

A: Get to Waypoint Alpha
B: Get to Waypoint Brav
C: Get to Waypoint Charl

malebolgia13 was tagged by jstevec's grenade.
jstevec is joining the server.

malebolgia13 is joining the server.

Figure 19. Screen Shot 18


(Unofficial Server) 2.0.0

North West Objective Alpha Area

OBJECTIVES (F4 toggles)

A: Get to Waypoint Alpha
B: Get to Waypoint Brav
C: Get to Waypoint Charl

malebolgia13 was tagged by jstevec's grenade.
jstevec is joining the server.

malebolgia13 is joining the server.

Figure 20. Screen Shot 18 with false coloring

Source	Time-independent Probability of Detection	Probability of Detection within 5 Seconds
Experiment	0.2143	0.0000
ACQUIRE-like algorithm	0.9990	0.9323
Color-modified ACQUIRE	0.9990	0.9324
Color Histograms	0.9997	0.9469

```
Processing Frames . . .

Using the ACQUIRE-like algorithm
       Ave. Target Pix Val: 72.9273  Ave. Bckgrd Pix Val: 102.779
       Contrast Value:      0.496533
       n:                   3.68239
       e:                   5.27767
pdet: 0.998973
p(5.0): 0.932347

Using the color-aware ACQUIRE-like algorithm
       Ave. Target Red Val: 74.1455  Ave. Bckgrd Red Val: 104.357
       Ave. Target Grn Val: 73.5818  Ave. Bckgrd Grn Val: 103.411
       Ave. Target Blu Val: 71.0545  Ave. Bckgrd Blu Val: 100.569
       Contrast Value:      0.496572
       n:                   3.68268
       e:                   5.27787
pdet: 0.998973
p(5.0): 0.932362

Using color histograms

Target Red:   [0      1       54      0       0       0       0       0]
Bkgrnd Red:   [0      2       120     174     57      0       0       0]
Target Green: [0      3       52      0       0       0       0       0]
Bkgrnd Green: [0      4       122     176     51      0       0       0]
Target Blue:  [0      8       47      0       0       0       0       0]
Bkgrnd Blue:  [0      6       130     202     15      0       0       0]
       L1 diff:             0.573654
       n:                   4.25433
       e:                   5.67803
pdet: 0.999731
p(5.0): 0.946942
```

Table 5. Detection probability comparison table and program output for Screen Shot 18

The results from our algorithms versus the experimental data for Screen Shot 18, depicted in Figure 19, are less encouraging. Recall that ACQUIRE bases its detection probabilities on two main inputs: The target vs. background contrast and the size of the target within the visual field of the observer. For Screen Shot 18, the contrast between the target and the background is significant. It appears that many of the experimental subjects had problems with Screen Shot 18. Two reasons for this are apparent: (1) the target is in clutter, and (2) the target's body position is such that it is inline with the clutter. In effect, the target looks like it is part of the tree. ACQUIRE does not deal with either of these types of concealment, nor do either of our modified algorithms. Again we suggest that the field of computer vision may have the solution to this non-trivial problem. Specifically, calculations that deal with texture would be helpful in this case.

6. Screen Shot 25

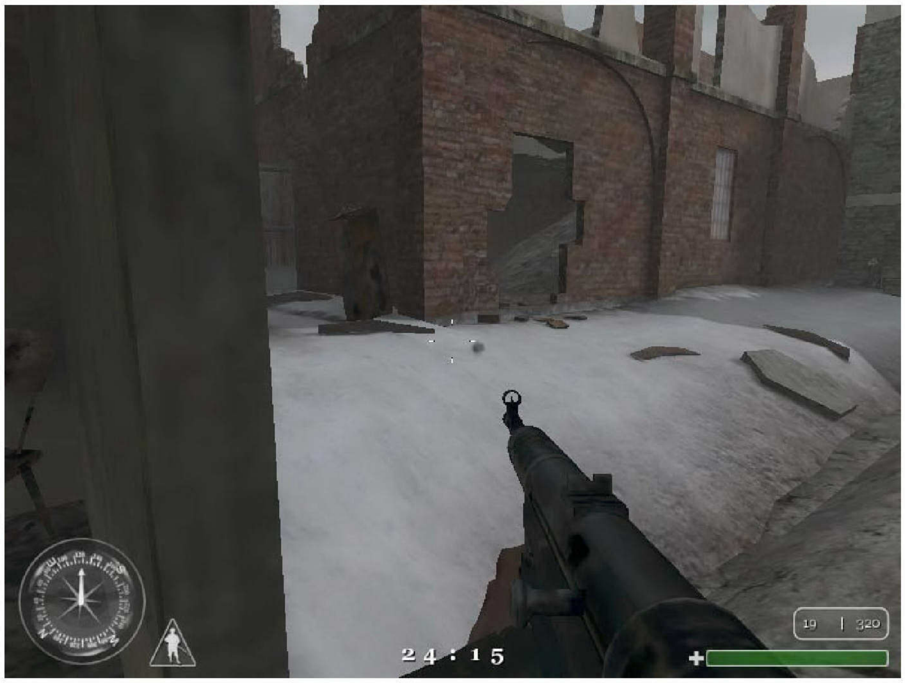


Figure 21. Screen Shot 25

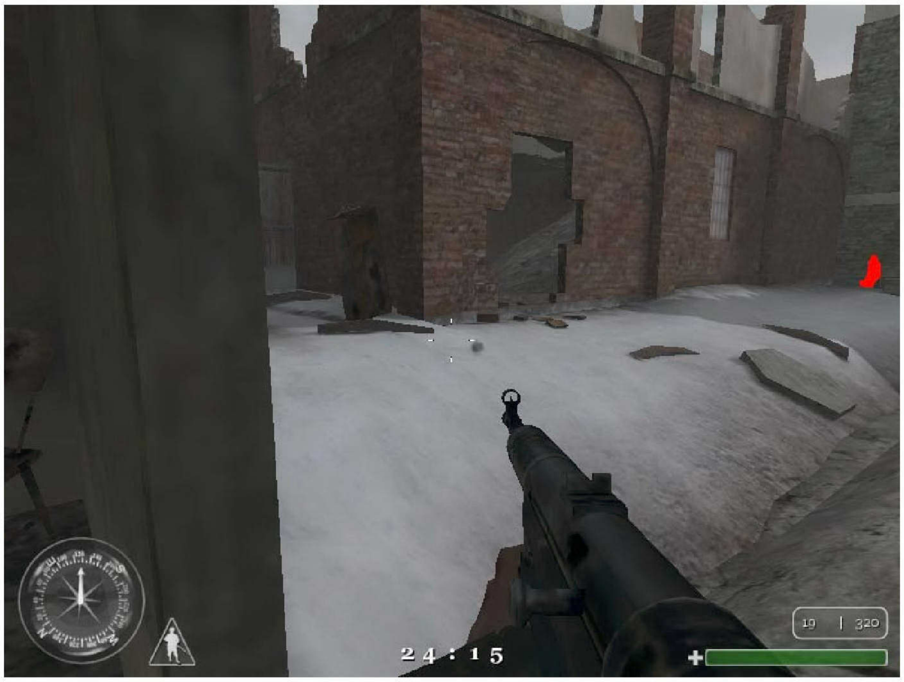


Figure 22. Screen Shot 25 with false coloring

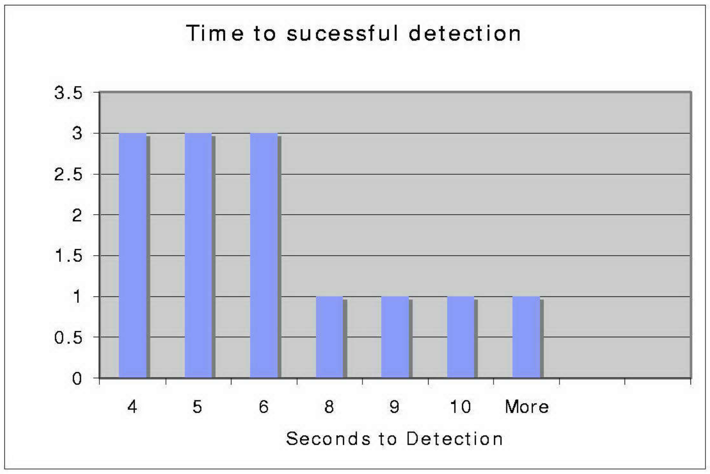


Figure 23. Distribution of successful experimental detection times for Screen Shot 25

Source	Time-independent Probability of Detection	Probability of Detection within 5 Seconds
Experiment	0.9286	0.4286
ACQUIRE-like algorithm	0.5829	0.3234
Color-modified ACQUIRE	0.5796	0.3209
Color Histograms	0.1845	0.0678

```
Processing Frames . . .

Using the ACQUIRE-like algorithm
        Ave. Target Pix Val: 38.466   Ave. Bckgrd Pix Val: 39.72
        Contrast Value:      0.0621438
        n:                   1.10119
        e:                   3.47083
pdet: 0.582869
p(5.0): 0.323497

Using the color-aware ACQUIRE-like algorithm
        Ave. Target Red Val: 39.4204  Ave. Bckgrd Red Val: 40.2182
        Ave. Target Grn Val: 38.6529  Ave. Bckgrd Grn Val: 40.4909
        Ave. Target Blu Val: 37.3248  Ave. Bckgrd Blu Val: 38.4509
        Contrast Value:      0.0619072
        n:                   1.097
        e:                   3.4679
pdet: 0.57958
p(5.0): 0.320875

Using color histograms
Target Red:   [13      298     3       0       0       0       0       0]
Bkgrnd Red:   [18      251     6       0       0       0       0       0]
Target Green: [16      297     1       0       0       0       0       0]
Bkgrnd Green: [17      252     6       0       0       0       0       0]
Target Blue:  [24      289     1       0       0       0       0       0]
Bkgrnd Blue:  [32      239     4       0       0       0       0       0]
        L1 diff:             0.0351314
        n:                   0.62253
        e:                   3.13577
pdet: 0.184487
p(5.0): 0.0677602
```

Table 6. Detection probability comparison table and program output for Screen Shot 25

Screen Shot 25, depicted in Figure 21, was another image where our algorithm failed to replicate the results found in the experiment. From the contrast calculations one can see that there is a very low level of contrast between the target and the background. Why was the detection rate so high in the experiment? There are several possible reasons. First, the viable hiding places in this scene are limited. One can cycle though these places quickly and, thus, take a closer look at each. Also, the target appears near the edge of the scene. We suspect that this increases the likelihood of detection.

Of additional interest is the large variance in the detection times of the experimental subjects. This indicates that Screen Shot 25 was, in fact, a difficult scene for the subjects. However, the subjects were told that there was likely a target in the scene and so they were less likely to give up. This undoubtedly adds to the high detection percentages.

7. Screen Shot 37

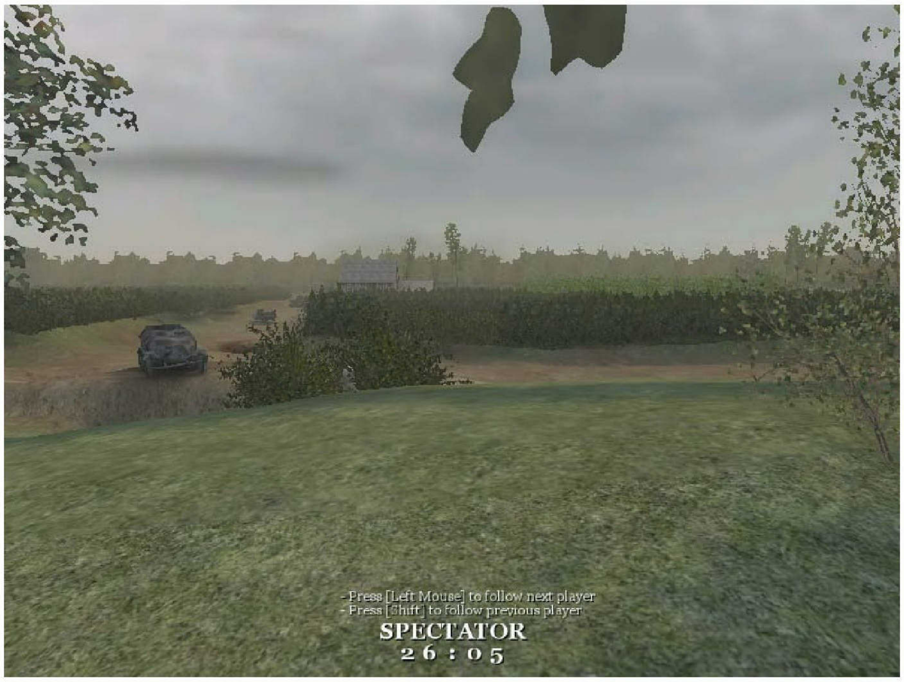

- Press [Left Mouse] to follow next player
- Press [Shift] to follow previous player
SPECTATOR
26 : 05

Figure 24. Screen Shot 37

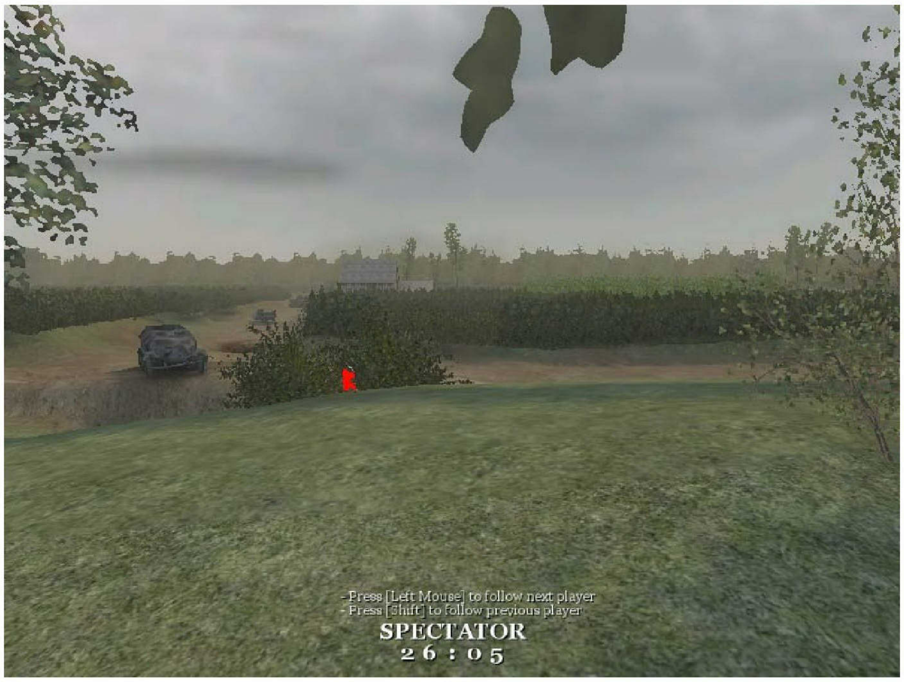

- Press [Left Mouse] to follow next player
- Press [Shift] to follow previous player
SPECTATOR
26 : 05

Figure 25. Screen Shot 37 with false coloring

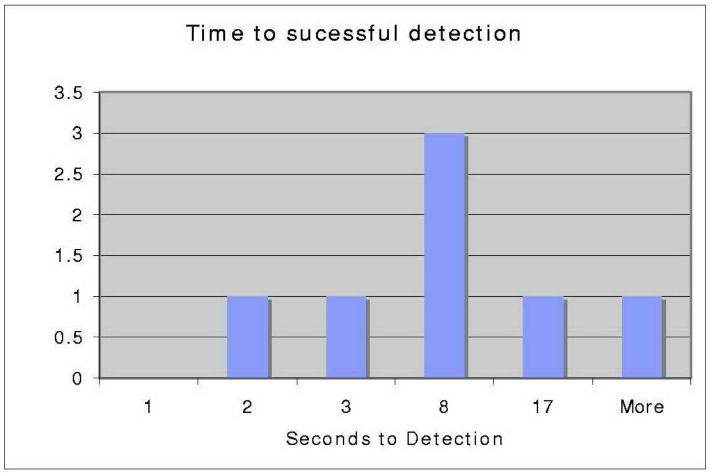


Figure 26. Distribution of successful experimental detection times for Screen Shot 37

Source	Time-independent Probability of Detection	Probability of Detection within 5 Seconds
Experiment	0.5000	0.1429
ACQUIRE-like algorithm	1.0000	0.9472
Color-modified ACQUIRE	1.0000	0.9472
Color Histograms	0.9999	0.9472

```
Processing Frames . . .

Using the ACQUIRE-like algorithm
       Ave. Target Pix Val: 63.8151  Ave. Bckgrd Pix Val: 39.9457
       Contrast Value:      1.55216
       n:                   19.3242
       e:                   16.2269
pdet: 1
p(5.0): 0.947196

Using the color-aware ACQUIRE-like algorithm
       Ave. Target Red Val: 66.5613  Ave. Bckgrd Red Val: 42.837
       Ave. Target Grn Val: 66.8774  Ave. Bckgrd Grn Val: 44.25
       Ave. Target Blu Val: 58.0065  Ave. Bckgrd Blu Val: 32.75
       Contrast Value:      1.6119
       n:                   20.0679
       e:                   16.7476
pdet: 1
p(5.0): 0.947196

Using color histograms

Target Red:   [6      51      94      4      0      0      0      0]
Bkgrnd Red:   [30     44      15      3      0      0      0      0]
Target Green: [5      54      91      5      0      0      0      0]
Bkgrnd Green: [23     52      13      4      0      0      0      0]
Target Blue:  [12     88      54      1      0      0      0      0]
Bkgrnd Blue:  [63     20      8       1      0      0      0      0]
       L1 diff:             0.45029
       n:                   5.60612
       e:                   6.62429
pdet: 0.999989
p(5.0): 0.947186
```

Table 7. Detection probability comparison table and program output for Screen Shot 37

Screen Shot 37, depicted in figure 24, is an example of a scene where the contrast between the target and the background is substantial, yet the experimental detection rate was low compared to our algorithms. Note that the edges of the target are obstructed from view, in effect masking the human-like aspects of the target and, thus, making the target difficult to detect, despite the contrast. The amount of clutter in the scene adds to the confusion. Additionally, colors similar to the targets colors appear elsewhere in the scene. Our ACQUIRE-like algorithms take none of these issues into account.

D. CONCLUSIONS AND SUGGESTED FUTURE WORK

Little is known about how well perception in 3-D virtual environments, either by algorithms or by humans looking at screens or with head mounted displays (HMDs), actually corresponds to perception in the real world. There is simply no data on this subject that we are aware of. Consider if you came across the scene in Figure 13 in real life. It is likely that you would detect the target in this situation, probably rather quickly. There are several reasons for this. First are the benefits that come from 3-dimensional eyesight. The target would stand out from the wall, unless he was in deep shadow at which the eye becomes less effective. Second, we know that movement plays a large role in target detection. So, if the target moved at all we would likely detect him as well. Reece and Wirthlan address motion, specifically, making the probability of detection higher for a moving target, but there are still a number of issues to address in this line of research, including that, in some cases motion might mask detection. Consider a soldier moving in a forest with a myriad of wind blown leaves or a soldier moving through swirling smoke. Improved algorithms will have a positive impact on gaming and military training simulation emersion and improve the validity of military simulations.

We have only scratched the surface of what can be done with image inputs to perception algorithms. However, we have shown that such images can be used as inputs for these algorithms and that such algorithms can outperform line-of-sight traces. The fields of computer vision and image processing undoubtedly have a great deal to add to this line of research. Further development of algorithms using these techniques is encouraged.

LIST OF REFERENCES

Bererton04 — Bererton, Curt. *State Estimation for Game AI Using Particle Filters*. AAAI. 2004.

Blumberg96 — Blumberg, Bruce Mitchell. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. PhD Dissertation, MIT. 1996.

Blumberg97 — Blumberg, Bruce Mitchell. *Go with the Flow: Synthetic Vision for Autonomous Animated Creatures*. MIT Media Lab. 1997.

Champion99 — Champion, D., Fatale, L., and Krause, P. Department of the Army. *Effects of Vegetation on Line-of-Sight (LOS) for Dismounted Infantry Operations* (TRAC-WSMR-TR-99-001(R). June 1999.

Darken04 — Darken, Christian J. *Visibility and Concealment Algorithms for 3D Simulations*. Behavior Representation in Modeling and Simulation Conference. 2004.

Delta3D — Delta 3D Open Source Gaming and Simulation Engine. Accessed on 25 March 2005 at http://delta3d.org/.

Forsyth03 — Forsyth, David A. and Ponce, Jean. Computer Vision: A Modern Approach. Prentice-Hall. 2003.

GP2_04 — Lastra, A., Lin, M., and Manocha, D. (Editors). *2004 ACM Workshop on General-Purpose Computing on Graphics Processors*. Organizing Committee of the 2004 ACM Workshop on General-Purpose Computing on Graphics Processors. 2004

Isla02 — Isla, Damian A. and Blumberg, Bruce M. *Object Persistence for Synthetic Creatures*. MIT Media Lab. 2002.

Johnson58 — Johnson, John. *Analysis of Image Forming Systems*. Image Intensifier Symposium (AD220160). October 1958.

Laird00 — Laird, John E. *An Exploration into Computer Games and Computer Generated Forces*. University of Michigan. 2000.

Leonard03 — Leonard, Tom. *Building an AI Sensory System*. Game Developer's Conference. 2003.

Lind95	Lind, Judith. *Target Acquisition Models for* Janus (A), NAWCWPNS TM 7811, Naval Air Warfare Center Weapons Division. 1995.
Lind95_2	Lind, Judith. *Searching and Scanning: A Review of Lawrence W. Stark's Vision Models.* Naval Postgraduate School. 1995.
Lind95_3	Lind, Judith. *Soviet Visual Perception Research: Application to Target Acquisition Modeling.* Naval Postgraduate School. 1995.
Mitchell97	Mitchell, Tom M. Machine Learning. McGraw-Hill. 1997.
Morgan03	Morgan, D., "Algorithmic Approaches to Finding Cover in Three-Dimensional Virtual Environments", Master's Thesis, Naval Postgraduate School. 2003. Accessed on 25 March 2005 at http://www.movesinstitute.org.
Peri95	Peri, Eli. Vision Models for Target Detection and Recognition. World Scientific. 1995.
Pursel04	Pursel, Eugene Ray. *Synthetic Vision: Visual Perception for Computer Generated Forces Using the Programmable Graphics Pipeline.* Naval Postgraduate School Masters Thesis. 2004. Accessed on 25 March 2005 at http://www.movesinstitute.org.
Reece96	Reece, Douglas A. and Wirthlan, Ralph. *Detection Models for Computer Generated Individual Combatants.* Proceeding of the 6th Conference on Computer Generated Forces and Behavioral Representation. 1996.
Russell03	Russell, Stuart J. and Norvig, Peter. Artificial Intelligence: A Modern Approach (Second Edition). Prentice Hall. 2003.
Wickens04	Wickens, Christopher D., et al. An Introduction to Human Factors Engineering. Pearson Education. 2004.
NVESD1	U.S. Army Night Vision & Electronic Sensors Directorate. Standard Category ACQUIRE Approved Standard.
NVESD2	U.S. Army Night Vision & Electronic Sensors Directorate. Standard Category ACQUIRE Contrast Model.

INITIAL DISTRIBUTION LIST

1. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

2. Defense Technical Information
   Ft. Belvoir, VA 22060-6218

3. Anthony Ciavarelli
   Naval Postgraduate School
   Monterey, California